Advanced Number Plate Recognition for Speed Violation Detection: A Systematic Study and Implementation

Author: Hayk Hovhannisyan
BS in Data Science,
College of Science and Engineering (CSE)
American University of Armenia
Yerevan, Armenia
hayk_hovhannisyan@edu.aua.am

Supervisor: Dr. Davit Ghazaryan
College of Science and Engineering (CSE)
American University of Armenia
Yerevan, Armenia
dghazaryan@aua.am

***Abstract*—This paper presents the design and implementation of a vehicle speed detection system using footage captured from 2 or more cameras. These devices are placed at fixed locations with known distance between them, enabling the calculation of vehicle speeds based on the time taken for a vehicle to traverse the known distance. The system employs Automatic number plate recognition (ANPR) techniques such as You Only Look Once (YOLO) object detection algorithm and Optical Character Recognition (OCR) for text extraction. Later are used to identify vehicles, measure the time taken for traversal, and calculate average speeds using the formula $\mathrm{V} = \mathrm{D}/\mathrm{t}$. Additionally, a database is utilized to manage all vehicles' license plate recognition data.**

***Index Terms*—Vehicle Speed Detection, Real-time Data Management, You Only Look Once, Video Processing, Speed Violation Detection, ANPR, Real-time Object Detection**

I. Introduction

Automatic Number Plate Recognition (ANPR) is a technology that has gained significant traction due to its diverse range of use cases across various industries. From enhancing security and surveillance to streamlining traffic management and toll collection, ANPR systems have become integral components of modern-day infrastructure.

The motivation for implementing a Speed Violation detection system using ANPR in Armenia stems from the idea that the system could provide significant advantages over existing traffic monitoring methods, potentially improving law enforcement and addressing traffic violations more effectively. With similar systems already successfully implemented in numerous other countries, there is a strong case for adopting this technology in Armenia to improve road safety, reduce crimes related to vehicles, and enhance overall traffic management efficiency. As a final product, this technology could also be suggested to other countries, following successful implementations in various nations, to further enhance road safety and reduce crimes related to vehicles on a global scale.

II. System dataflow

The system operates by positioning two cameras at a fixed distance of 2950 meters from each other. The number of cameras is not limited to two; it can be extended to as many as needed. These cameras capture images of vehicles and send the data over the network using User Datagram Protocol (UDP) stream which is a type of network Transmission Control Protocol /Internet Protocol (TCP/IP) that is widely used for streaming video. Within the central processing station, the system employs two threads to continuously read the video stream. For each frame, real-time ANPR is performed using state-of the art YOLO (You Only Look Once) algorithm, specifically v8 developed in 2023. This process happens simultaneously in two threads. The system dynamically updates and stores relevant information such as detection time and location of vehicle license plates.

This process is illustrated in "Fig. 1",

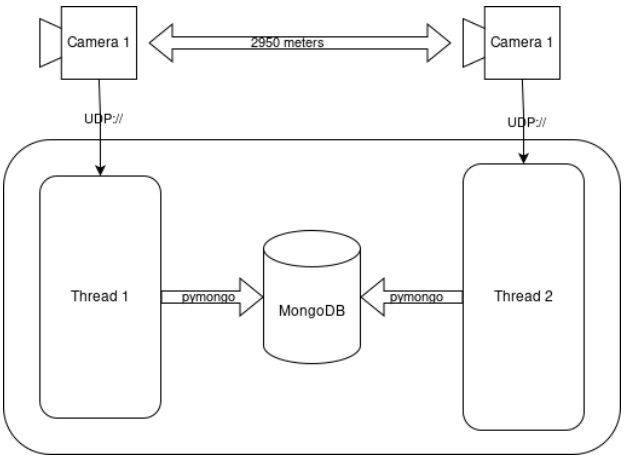


Fig. 1: System Architecture

III. Tools

The following subsections detail the tools utilized in the development and implementation of the ANPR system, each playing a crucial role in different stages of the process.

A. CVAT.ai

This online tool was used to annotate, label training images and then convert annotations into YOLO format, streamlining the annotation process for training the license plate detection system. [11]

B. Ultralytics

This Python library facilitated the training of image data using the YOLOv8n model, contributing to accurate license plate detection in the ANPR system. [12]

C. OpenCV

The Python library of OpenCV was used to manipulate video frames and optimize them for Optical Character Recognition (OCR). [13]

D. EasyOCR

EasyOCR was integrated into the system and configured under system requirements for optimal OCR. [14]

E. MongoDB

MongoDB served as the data storage solution for managing and storing detected license plate numbers along with their associated metadata. [15]

F. Google's Colaboratory/Jupyter Notebook

Google's Colaboratory, coupled with the NVIDIA Tesla T4 16G GPU, was used for training the model, leveraging its computational power to accelerate training speed and optimization [16]. Additionally, Jupyter Notebook was utilized for performance analysis of the system [17].

IV. Literature Overview

A. YOLO

The decision to use the YOLO architecture for the ANPR system was based on its fast and accurate object detection abilities, crucial for real-time data processing. YOLO distinguishes itself by having a one-stage prediction algorithm [5]. YOLO utilizes a single end-to-end convolutional neural network that processes RGB images in a single forward propagation pass, examines the image only once and makes predictions. Studies comparing YOLO with other models such as Single Shot Detectors (SSD) [4] and Region-Based Convolutional Neural Networks (R-CNN) [3] have shown its speed and accuracy advantages [1].

1) YOLO Architecture: YOLO operates by dividing the input image into a grid and predicting bounding boxes and class probabilities for each grid cell. The algorithm works as follows:

- Divide the input image into an S × S grid.
- For each grid cell, predict B bounding boxes and confidence scores.
- Apply non-maximum suppression (NMS) to filter out overlapping bounding boxes.
- Output the final bounding boxes along with their class probabilities.

The whole inference process is depicted in "Fig. 2"

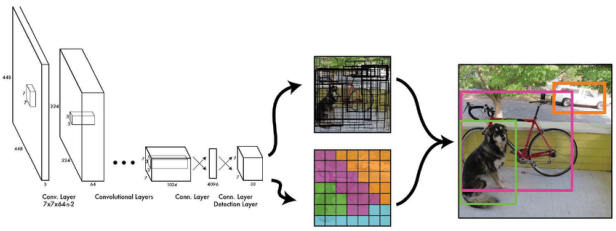


Fig. 2: YOLO algorythm pipeline [6]

B. EasyOCR Architecture

EasyOCR is an Optical Character Recognition (OCR) tool that utilizes deep learning models to recognize text from images. It employs Convolutional Neural Networks (CNNs) for text detection and recognition tasks. The architecture involves:

- Preprocessing the input image to enhance text visibility.
- Utilizing a CNN-based detector to locate text regions in the image.
- Employing a text recognizer network to convert detected text regions into machine-readable text.
- Post-processing steps for refining text recognition results. [7]

EasyOCR is known for its simplicity and ease of use, making it a popular choice for various text recognition applications.

This combination of YOLO for object detection and EasyOCR for text extraction forms a robust solution for the ANPR system, ensuring efficient and accurate processing of data in real-time scenarios.

V. Data

For the system implementation, two types of data were used.

A. Video Footage

Two video footage clips, each lasting 5 minutes, were captured simultaneously using a smartphone at a resolution of 3840 px × 2160 px (8.3 MP) and a frame rate of 30 FPS. This setup was designed to simulate real-time scenarios and assess the system's performance under such conditions. An example frame from the footage is shown in Figure 3.

B. License Plate Dataset

The Croatian license plate dataset was used in the ANPR system [8]. The data was formatted according to the YOLOv8 format. In addition to this dataset, an Armenian license plate dataset was scraped. However, despite its availability, the Armenian dataset was not integrated into the training process. This decision stemmed from the system's already robust performance and saturation, making the inclusion of the Armenian dataset unnecessary for achieving accurate results. It's worth noting that the Armenian dataset would remain available and can be hosted for future reference or potential use in other projects.

Fig. 3: Example from footage. The image intentionally omits certain numbers for security reasons.

1) Image: Each image in the dataset is a $w \times h \times 3$ dimensional vector as w, h stand for image height, width respectively. 3 is represents the number of color channels of the RGB image. The images are annotated to includes a bounding box representing the license plate. The example is illustrated in Fig 4

Fig. 4: Annotated image example

2) Label: The label format consists of tabular data with seven columns:

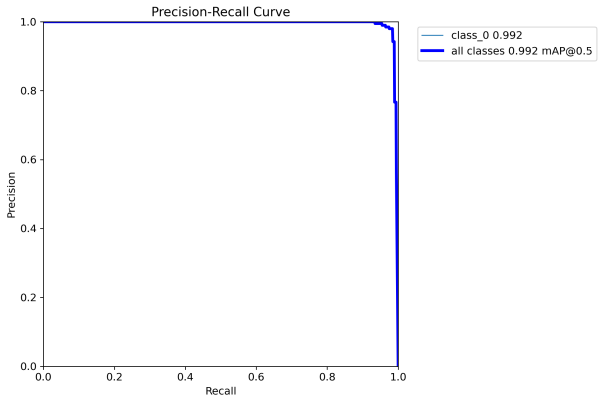
- Bounding Box Center X Coordinate (B_x)
- Bounding Box Center Y Coordinate (B_y)
- Width of the Bounding Box (B_w)
- Height of the Bounding Box (B_h)
- Class Confidence Scores (C_1,C_2, ..., C_n)

The label vector is illustrated below:

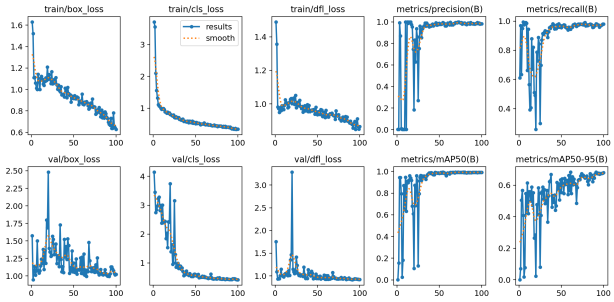
$$[B_x, B_y, B_w, B_h, C_1, C_2, ..., C_3]$$

C. Training Process

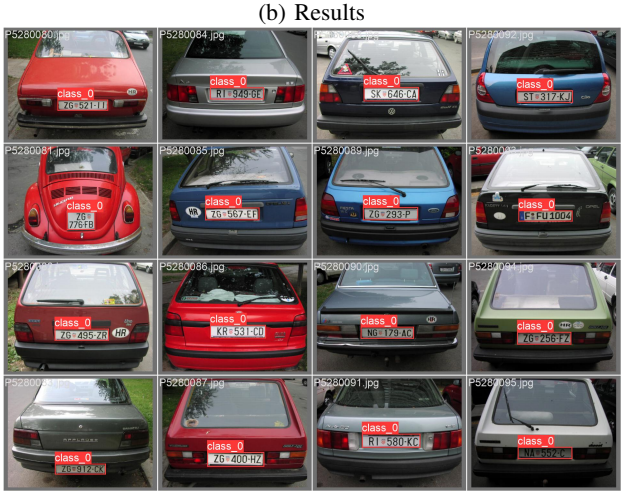
The training process follows a supervised learning approach. Each image vector (X_{train}) is paired with its corresponding label vector (Y_{train}). These pairs are fed into the YOLO CNN for training, iteratively processing each image in the training set. The details of training results can be observed in Fig. 5.



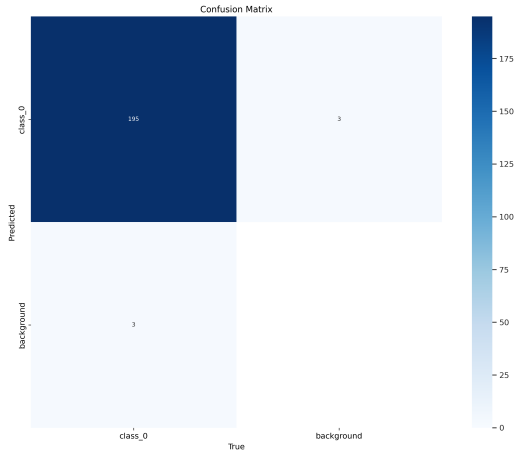
(a) Precision Recall Curve



(b) Results



(c) Validation on Batch of Images



(d) Confusion matrix

Fig. 5: Train results on the Plates Dataset

VI. Video Processing Pipeline

The video processing occurs in multiple stages, analyzing each frame of the video footage to generate text predictions if a license plate is visible within the frame.

A. Vehicle Localization

The system uses pre-trained weights from yolov8n.pt to detect passing vehicles in any frame of the video. The 'n' in 'yolov8n' denotes a nano model chosen for its efficient inference results without compromising computational time [2].

B. License Plate Detection

After localizing vehicles, the system focuses on the region of interest (ROI) and employs a second YOLO model with plate.pt weights specifically trained for license plate detection.

C. Image Processing Pipeline

The cropped license plate undergoes additional processing before OCR can efficiently work on it. This processing pipeline includes the following steps:

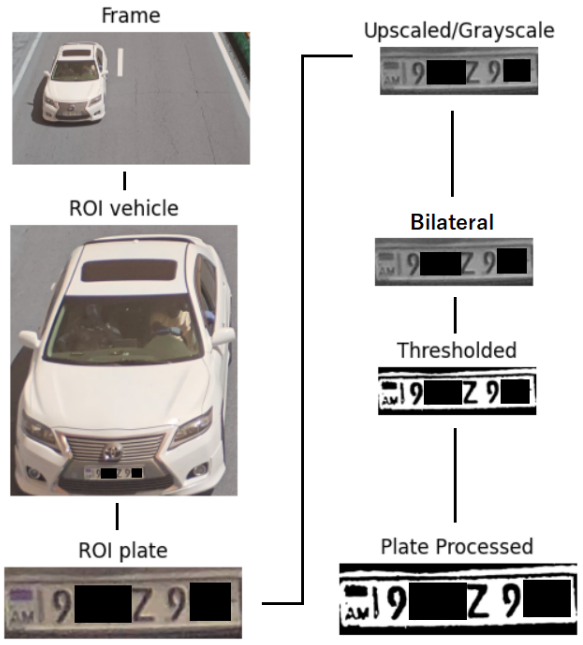


Fig. 6: Annotated image example

- **Upscaling the image:** The dimensions of the image are artificially increased to preserve the features of characters for OCR.
- **Converting the image to grayscale:** This is a necessary step when applying OCR, as it simplifies the image to a single channel representing the intensity of each pixel.
- **Applying a bilateral filter:** This filter removes noise from the data while preserving edges. Unlike a Gaussian blur, which smooths the entire image uniformly, the bilateral filter considers both color similarity and spatial proximity of pixels. Pixels with similar colors and close spatial distances are weighted more heavily, preserving sharp edges in the image while smoothing out noise [9].
- **Thresholding the image to create a binary image:** In this step, a threshold value is chosen, and pixel values above this threshold are set to a maximum value (typically 255 in grayscale), while pixel values below the threshold are set to a minimum value (usually 0). This creates a binary image where pixel values are either black or white, simplifying the image for further processing.

The Fig. 6 shows the working pipeline.

D. Optical Character Recognition (OCR)

The processed image is then passed to an OCR system, which makes predictions based on the text it detects in the processed license plate image. The system processes multiple frames as the vehicle moves, performing OCR detection on each frame. If OCR produces an incorrect character guess, the system disregards it and retrieves the correct input from another frame. Once the vehicle has passed, the system analyzes all character guesses for each position across frames and selects the most frequently occurring character, ultimately returning the accurate license plate information.

VII. Data Management

A. Data Flow and Formating

Once data is generated, it is routed to temporary storage, where images are stored temporarily for every frame. This temporary storage mechanism ensures that each frame's information is preserved until processing for each particular vehicle is complete.

Detail illustraition of the process in Fig 7

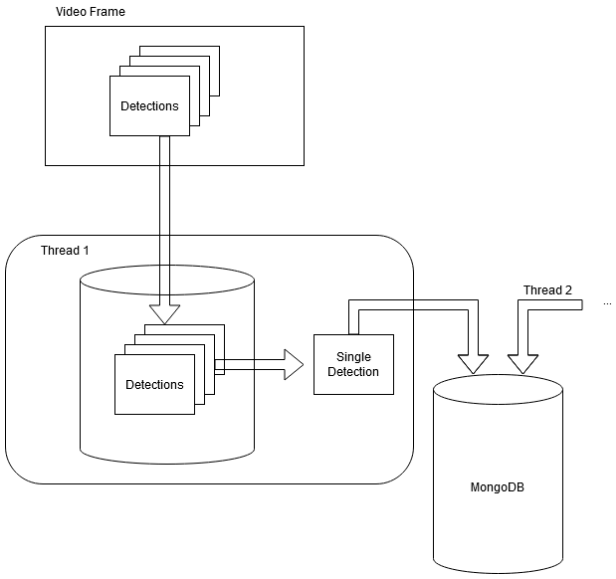


Fig. 7: Data Management Process

When an image disappears from a frame, a listener script is triggered to retrieve information from the temporary storage.

This information typically includes the datetime of the event, the detected license plate number, the path to the stored image, and the source camera identifier. This data is then inserted into the database, maintaining a structured record of events and observations.

This process is replicated for the second video footage, running concurrently to simulate a real-time scenario effectively. The parallel handling of data from multiple sources ensures that the system remains responsive and capable of managing continuous streams of information. The final data that appears in the MongoDB has following structure:

ID	datetime	image_path	source
...	...	...	...
00AB123	2024-05-09 12:00:01	/path/to/img	1
...	...	...	...
00AB123	2024-05-09 12:00:25	/path/to/img	2
...	...	...	...

B. Violation Detection

The detection algorithm operates on the main database, focusing on calculating the time gap between data collected from the initial and subsequent video feeds. This time difference computation is essential for estimating the vehicle's speed accurately. After velocity estimation, the algorithm assesses whether a violation has occurred based on the calculated speed compared to predefined speed limit thresholds.

VIII. Results

The results of the project indicate significant progress in Automatic Number Plate Recognition (ANPR) capabilities. However, certain challenges remain to be addressed for optimal performance.

A. OCR Recognition Efficiency

The Optical Character Recognition (OCR) component, while functional, exhibits inefficiencies that impact overall system performance. Improving the OCR algorithm's efficiency is crucial for enhancing the ANPR system's accuracy and responsiveness.

B. Implementation Status

It's important to note that the current implementation is not final but has demonstrated promising results. Further optimizations and refinements are underway to address the identified issues and enhance system performance.

C. Processed Result

Figure 8 showcases one of the processed results, highlighting the ANPR system's ability to accurately detect and recognize license plate numbers in real-world scenarios.

Fig. 8: Processed Result Example
No Violation: Threshold 90km/h: Registered: 67.22 km/h

IX. Future Work and Improvements

This milestone marks a significant step in the ongoing development of the process. While the system demonstrates functionality and addresses the primary challenge, it falls short of achieving optimal performance. There are several key areas that have been identified for further enhancement:

1) **Optimizing System Resources**: Continued efforts will focus on optimizing system resources to ensure efficient use of computing power and storage capacity.
2) **Data Redundancy Optimization**: A priority will be to minimize redundant data and streamline data storage and processing workflows for improved system performance.
3) **Reducing Data Gap**: Efforts will be made to shorten the time gap between data appearing in the database, potentially eliminating the need for temporary dataframes and enhancing real-time data processing capabilities.
4) **Blur Solution**: Addressing issues related to image blur through advanced techniques such as Weiner deconvolution [10] will be explored to improve the accuracy of vehicle detection and recognition.

X. Conclusion

Our ANPR system, implemented using YOLO (You Only Look Once) and OCR (Optical Character Recognition), leverages real-time database management for efficient operations. These advancements have significantly enhanced the system's functionality and performance.

Acknowledgment

I would like to express my sincere gratitude to my capstone supervisor, Dr. Davit Ghazaryan, for his invaluable guidance, support, and expertise throughout the process of developing this capstone project. His insights and feedback have been instrumental in shaping the project's direction and ensuring its successful completion. I am grateful for the opportunity to work under his mentorship and for his unwavering support during this journey.

References

[1] A. John and D. D. Meva, "A Comparative Study of Various Object Detection Algorithms and Performance Analysis", *INTERNATIONAL JOURNAL OF COMPUTER SCIENCES AND ENGINEERING*, vol. 8, pp. 158–163, 10 2020.

[2] I. Lazarevich, M. Grimaldi, R. Kumar, S. Mitra, S. Khan, and S. Sah, "YOLOBench: Benchmarking Efficient Object Detectors on embedded Systems". 7 2023. https://arxiv.org/abs/2307.13901

[3] R. Girshick, J. Donahue, T. Darrell, and J. Malik, 'Rich feature hierarchies for accurate object detection and semantic segmentation'. 11 2013. https://arxiv.org/abs/1311.2524
[4] W. Liu et al., SSD: Single Shot MultiBox Detector. 1 2016, pp. 21–37. https://arxiv.org/abs/1512.02325
[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, 'You only look once: Unified, Real-Time Object Detection'. 6 2015. https://arxiv.org/abs/1506.02640
[6] D. Tran, P. Fischer, A. Smajic, and Y. So, 'Real-time Object Detection for Autonomous Driving using Deep Learning', 03 2021. http://dx.doi.org/10.13140/RG.2.2.19546.26562
[7] C. K. Gomathy, 'OPTICAL CHARACTER RECOGNITION', May 2022.
[8] 'Projekt "License Plates"'. 2003. https://www.zemris.fer.hr/projects/LicensePlates/english/
[9] S. Paris, P. Kornprobst, J. Tumblin, and F. Durand, 'Bilateral Filtering: Theory and applications', Foundations and trends in computer graphics and vision, vol. 4, no. 1, pp. 1–75, 1 2008
[10] J. Dong, S. Roth, and B. Schiele, 'Deep Wiener Deconvolution: Wiener meets Deep Learning for image deblurring'. 3 2021. https://arxiv.org/abs/2103.09962
[11] CVAT.AI https://www.cvat.ai/
[12] Ultralytics. https://docs.ultralytics.com/
[13] OpenCV. https://opencv.org/
[14] Jaided.AI, EasyOCR. https://www.jaided.ai/easyocr/
[15] MongoDB https://www.mongodb.com/
[16] Google's Colaboratory https://colab.research.google.com/
[17] Jupyter https://jupyter.org/